1. To answer this question, we will create 3 vectors named *first, second*, and *third*.

   (a) Create the vectors first, second, and third. Each vector should be made up of the following elements: first=(6,10,9,10,8,4,9,1,3,6), second=(4,2,3,2,2,2,5,5,1,2), third=(12,12,13,11,13,10). Make sure to print your vectors to make sure that you have entered the correct values in the correct locations.

   Solutions:

   ```
   >first=c(6,10,9,10,8,4,9,1,3,6)
   > second=c(4,2,3,2,2,2,5,5,1,2)
   >third=c(12,12,13,11,13,10)
   ```

   (b) Try adding the vectors *first* and *second* together. Is the result what you expect?

   Solutions:

   ```
   first+second
    [1] 10 12 12 12 10  6 14  6  4  8
   ```

   (c) Now try adding the vectors *first* and *third* together. What is the length of *third* compared to *first*? How were we able to add these two vectors together?

   Solutions:

   ```
   first+third
    [1] 18 22 22 21 21 14 21 13 16 17
   Warning message:
   longer object length
   is not a multiple of shorter object length in: first + third
   ```

   The `first` vector is 10 elements long, and the `third` vector is 6 elements long. R will *recycle* values from the `third` vector so that it matches the length of `first`

   (d) Let's use some of the vector specific functions. What is the sum of the elements in each of the vectors? Add these sums together to get a grand total?

   Solutions:

   ```
   > sum(first)
   [1] 66
   > sum(second)
   [1] 28
   > sum(third)
   [1] 71
   > 66+28+71
   [1] 165
   or
   > sum(first)+sum(second)+sum(third)
   [1] 165
   ```

   (e) Feel free to play around with some of the functions for vectors, and how they can be combined into equations involving vectors.
   Solutions:
   Try some fancy stuff like the following!

   ```
   >sqrt(sum(first))+prod(second)
   [1] 9608.124
   ```

(f) Based on your experiences with vectors, can you compare and contrast using R and Excel to perform data calculations? For example, Excel allows you to perform calculations like =SUM(firstcell:lastcell) and others. How does performing this type of calculation in R compare to performing this type of calculation in Excel?
Solutions:
R can perform the same (and more) calculations than Excel. This means that if you are the sort of person who likes to calculate at least a few statistical estimates manually to make sure you understand where numbers are coming from, then you can do this in R. You can even check that R is doing things correctly! (it is)

2. Now we are going to play with different ways of creating vectors in R. See if you can predict what the following commands will do, and then check your answer using R.

(a) seq(from=1,to=10,by=2)
Solutions:

```
>  seq(from=1,to=10,by=2)
[1] 1 3 5 7 9
```

(b) seq(from=5,to=1,by=-1)
Solutions:

```
seq(from=5,to=1,by=-1)
[1] 5 4 3 2 1
```

(c) seq(from=1,to=5,length=11)
Solutions:

```
seq(from=1,to=5,length=11)
 [1] 1.0 1.4 1.8 2.2 2.6 3.0 3.4 3.8 4.2 4.6 5.0
```

(d) x=c(1,2,3), rep(x, times=3)
Solutions:

```
> rep(x, times=3)
[1] 1 2 3 1 2 3 1 2 3
```

(e) rep(x,each=3)
Solutions:

```
> rep(x,each=3)
[1] 1 1 1 2 2 2 3 3 3
```

(f) y=x, rep(x,times=y)
Solutions:

```
> rep(x,times=y)
[1] 1 2 2 3 3 3
```

3. Recall that we made some vectors named *first, second*, and *third* in question 1. We will be using them again here to answer parts of this exercise.

(a) What happens if we bind the vectors together using the command `c(first,second,third)`?
Solutions:

```
>c(first,second,third)
 [1]  6 10  9 10  8  4  9  1  3  6  4  2  3  2  2  2  5  5  1  2 12 12 13 11 13 10
```

We end up making one long vector that is made up of the elements of `first`,`second`, and `third`, in that order.

(b) We are going to use a new function that is very similar to `c()` in this part. This function is called `cbind()` which stands for *column bind*. Type the command `cbind(first,second)`. What does this create? What happens if we type `cbind(first,second,third)`? What do you think the error message means?
Solutions:

```
> cbind(first,second)
      first second
 [1,]     6      4
 [2,]    10      2
 [3,]     9      3
 [4,]    10      2
 [5,]     8      2
 [6,]     4      2
 [7,]     9      5
 [8,]     1      5
 [9,]     3      1
[10,]     6      2
```

This creates a table, where each column of the table is one of the vectors that I gave as arguments.

```
> cbind(first,second,third)
      first second third
 [1,]     6      4    12
 [2,]    10      2    12
 [3,]     9      3    13
 [4,]    10      2    11
 [5,]     8      2    13
 [6,]     4      2    10
 [7,]     9      5    12
 [8,]     1      5    12
 [9,]     3      1    13
[10,]     6      2    11
Warning message:
number of rows of result
is not a multiple of vector length (arg 3) in: cbind(1, first, second, third)
```

The error message is referring to the fact that the `third` vector is shorter than the other vectors, and R is warning the user that it has recycled values from the `third` vector to create the table.

(c) We are going to use a new function that is very similar to `c()` in this part. This function is called `rbind()` which stands for *row bind*. Type the command `rbind(first,second,third)`. How does this compare to what happened when we used `cbind()`? What happens if we type the command `rbind(first,second,5)`?

Solutions:

```
> rbind(first,second,third)
       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
first     6   10    9   10    8    4    9    1    3     6
second    4    2    3    2    2    2    5    5    1     2
third    12   12   13   11   13   10   12   12   13    11
Warning message:
number of columns of result
is not a multiple of vector length (arg 3) in: rbind(1, first, second, third)
```

This creates a table, where each *row* of the table is one of the vectors that I gave as arguments. Similar to the earlier question, the fact that the `third` vector is shorter results in a warning message from R.

4. Using what you know about the commands discussed in lecture and introduced above, we are going to create a data layout for an experimental scenario.

Imagine that we are sampling from 5 different landfill sites. We are planning to sample once a month for the months of June, July, August, and September (4 sampling times). In each sample, we are testing for 3 quantities of interest: dissolved solids; suspended solids; and phytoplankton levels. Create 3 vectors named *Site, Time*, and *Analyte*. These vectors should be constructed such that if we use the command `cbind(Site,Time,Analyte,NA)` it creates a table representing all of our sampling combinations and has a column of NA's to represent where we will fill in our observation values.

hint: There is more than one way to create this table that will work.

hint: It can be extremely useful to think about how the table should look using a pen and paper before using any commands in R.

Solutions:
Yet again, there is more than one approach to answering this question. As long as each observation can be uniquely identified then the table setup, and commands used to create it should be considered good!

```
> Site=rep(1:5,each=12)
> Time=rep(1:4,each=3,times=5)
> Analyte=rep(1:3,times=20)
> cbind(Site,Time,Analyte,NA)
      Site Time Analyte
 [1,]    1    1       1 NA
 [2,]    1    1       2 NA
 [3,]    1    1       3 NA
 [4,]    1    2       1 NA
 [5,]    1    2       2 NA
 [6,]    1    2       3 NA
 [7,]    1    3       1 NA
 [8,]    1    3       2 NA
 [9,]    1    3       3 NA
[10,]    1    4       1 NA
[11,]    1    4       2 NA
[12,]    1    4       3 NA
[13,]    2    1       1 NA
[14,]    2    1       2 NA
[15,]    2    1       3 NA
[16,]    2    2       1 NA
[17,]    2    2       2 NA
[18,]    2    2       3 NA
[19,]    2    3       1 NA
[20,]    2    3       2 NA
[21,]    2    3       3 NA
[22,]    2    4       1 NA
[23,]    2    4       2 NA
[24,]    2    4       3 NA
[25,]    3    1       1 NA
[26,]    3    1       2 NA
[27,]    3    1       3 NA
[28,]    3    2       1 NA
[29,]    3    2       2 NA
[30,]    3    2       3 NA

etc.
```

5. You are going to have to read some data into R in order to complete this question. Look for the data set called H20Qex.csv that has been provided for you in the workshop folder. Choose a name for this data, and read it into R using the `read.table()` function.

   (a) After reading in the data, test the command `dim(dataname)`. Write down the values that this function gives us. Solutions:
   Note that the file name is H2zero, not H2ooooh. A bit confusing, but you have to specify this correctly for things to read into R.

   ```
   > H2Odata=read.table("h:/H20Qex.csv",header=TRUE,sep=",")
   > dim(H2Odata)
   [1] 30  2
   ```

   The `dim()` function returns two numbers, in this case, 30 and 2.

(b) One of the observation values is missing. We don't want this missing observation to ruin our analysis, so please remove this row of data.

Solutions:
The tenth observation is missing.

```
#specifies removing the 10th row, all columns
> H2Odata=H2Odata[-10,]
```

(c) If we use the command `dim(dataname)` now, what values does this function give us? What does the change reflect about our data?

Solutions:

```
> dim(H2Odata)
[1] 29  2
```

The `dim()` function tells us how many rows and columns we have in our data. When we remove the tenth row, our row count goes down to 29. 2 reflects the number of columns in the data.

(d) Try typing `dim(dataname)[2]`. What does the returned value suggest about the type of object that is created by the `dim()` function?

Solutions:

```
> dim(H2Odata)[2]
[1] 2
```

The value returned is a 2. This is the number of columns that we have in our data set. Because we can access this value using square brackets indicates that the output returned by the `dim()` function is a vector object.

(e) Find the minimum, maximum, and mean of the *observation* variable?

Solutions:

```
#finds the minimum
> min(H2Odata$observation)
[1] 0.06697026
#finds the maximum observation
> max(H2Odata[,2])
[1] 15.78163
#finds the mean observation
> attach(H2Odata)
> mean(observation)
[1] 1.807490
>detach(H2Odata)
```

I have demonstrated 3 different ways to access the `observation` vector, but realistically I would just pick one method I liked and use it.

(f) More sensibly, find the mean observation for each of the different *analyte* categories

Solutions:
Again, there are going to be multiple approaches to this problem. They might make a vector of the locations of the values that are 1 (or 2 or 3), and use that to extract the correct values. Or they might use code more similar to what I use below.

```
#creates a data set containing only the first analyte
H2Oone=H2Odata[which(H2Odata$analyte==1),]
> mean(H2Oone$observation)
[1] 0.786627
H2O2=H2Odata[which(H2Odata$analyte==2),]
> mean(H2O2$observation)
[1] 1.166435
> H2O3=H2Odata[which(H2Odata$analyte==3),]
> mean(H2O3$observation)
[1] 3.367321
```

(g) The 28th observation is a mistake! It should be 1.578 rather than 15.578. Correct this and find a new mean for this analyte.

Solutions:
Again, there are going to be multiple approaches to this problem.

```
> H2Odata[28,2]=1.578189096
> H2Othree=H2Odata[which(H2Odata$analyte==3),]
> mean(H2Othree$observation)
[1] 3.346950
```

(h) It turns out that the lab that produced the results couldn't accurately detect values below 0.7. Values below this are considered below the *Method Detection Limit.* Create a vector of TRUE and FALSE called *Below* that indicates whether a given observation is below the detection limit (TRUE), or above the detection limit(FALSE). Bind this column to the data set.

Solutions:

```
> Below=H2Odata[,2]<0.7
> Below
 [1]  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE
[15] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
[29] FALSE

> NewH2O=cbind(H2Odata,Below)
>NewH2O
   analyte observation Below
1        1  0.53040402 TRUE
2        1  1.82071728  FALSE
3        1  0.37020873 TRUE

etc.
```

(i) Using R, see if you can find a way to determine the proportion of values that fall below the method detection limit.

hint: You can sum the T/F column

Solutions:
There is also more than one way to calculate the solution to this problem.

```
> proportion=sum(Below)/dim(H2Odata)[1]
> proportion
[1] 0.4137931
```

(j) What proportion of observations are censored in the 2nd analyte group?

Solutions:
There is also more than one way to calculate the solution to this problem.

```
> NewH2O2=NewH2O[which(NewH2O$analyte==2),]
> proportion2=sum(NewH2O2$Below)/dim(NewH2O2)[1]
> proportion2
[1] 0.4
```